Date: January 7, 1998 COST257TD(98)
Issue: 1

# Using Gibbs Sampler in Simulating Multiservice Loss Systems

Pasi Lassila [1] and Jorma Virtamo [2]

Helsinki University of Technology

**Abstract.** *In this article we consider the problem of calculating the blocking probabilities of calls in a multiservice network by using simulation. The traditional approaches suffer from the fact that the state space explosion inherent in the system causes their efficiency to decrease correspondingly. We develop a method that alleviates the effect of the state space explosion. The method is based on using the so called Gibbs sampler to generate a Markov chain with the desired stationary distribution. In particular, by making an additional "virtual" step from each state and calculating the expected contribution from this step analytically, we are able to collect information from a subset of the whole state space for each generated sample. This leads to a smaller variance of the estimate for a given computational effort.*

## 1 Introduction

Modern broadband networks have been designed to integrate several service types into the same network. On the call scale, the process describing the number of calls present in the network can be modelled by a loss system. Associated with each call is the route through the network and the bandwidth requirements on the links. When the call is offered but there is not enough bandwidth on all the links along the requested route, the call is blocked and lost. The specific quantities we are interested in are the blocking probabilities for each call.

This is a natural extension of the model for the traditional single service telephone network. The steady state distribution of the system has a well known product form. A problem with the exact solution is, however, that it requires the calculation of a so called normalization

[1] Pasi.Lassila@hut.fi

[2] Jorma.Virtamo@hut.fi

constant, which entails the calculation of a sum over the complete allowed state space of the system. In a practical size network with possibly hundreds of classes and high speed links, the state space very rapidly becomes astronomical. In fact, it is known that the calculation of the normalization constant is an NP complete problem [4, p. 230].

In such a situation we have two alternatives: to use analytical approximations or to simulate the problem to a desired level of accuracy. In this paper we will be dealing with efficient methods for doing the simulation. Traditionally the simulation approaches have focused on either Monte Carlo (MC) summation techniques or the Markov chain simulation techniques. MC summation has been extensively studied by Ross [4, chap. 6]. Markov chain simulation methods include the regenerative method, developed by Crane and Iglehart [1, 2], which has been lately used in the context of rare event simulation in loss networks by Heegaard [3].

The problem with the aforementioned methods is that they become computationally very intensive as the state space grows. The reason is that they collect information about the state space on a state-per-state basis and when the sheer size of the state space is the source of the problem, the methods are bound to fail at some point. What is actually needed is a method that would allow the aggregation of state space information, such that it would be possible to collect more information with each sample than just the information of the sample itself.

In this article, we will present a method that is based on a family of simulation methods called Markov Chain Monte Carlo (MCMC) methods using the Gibbs sampler [5]. This method enables us to exploit the product form solution of the system, and we are able to calculate part of the problem analytically while the simulation is being carried out. In practice, this means that with each generated sample we can collect information of not just the current sample state, but a large number of other surrounding states. However, it will be shown that actually the MCMC method for the sample generation is not crucial for the application of our improved sampling method and it can be used in the connection of the normal MC summation method as well.

The rest of the paper is organized as follows. Chapter 2 introduces the basic stochastic model of the problem. Chapter 3 describes the Gibbs sampling method for loss networks. Chapter 4 contains the main results of this paper with methods for improving the performance of the Gibbs sampler. Efficient implementation of the improved method is discussed in chapter 5 and the numerical results are presented in chapter 6 and the conclusions in chapter 7.

# 2 Model Description

Consider a network consisting of $J$ links, indexed with $j = 1, \ldots, J$, each having a capacity of $C_j$ resource units. The network supports $K$ classes of calls. Associated with a class-$k$ call, $k = 1, \ldots, K$, is an offered load $\rho_k$ and a bandwidth requirement of $b_{j,k}$ units on link $j$. Note that $b_{j,k} = 0$ when class-$k$ call does not use link $j$. Let the vector $\mathbf{b}_j = (b_{j,1}, \ldots, b_{j,K})$

denote the required bandwidths of the classes in the system on link $j$. Also, we assume that a call is always accepted if there is enough capacity left and that the blocked calls are cleared. The state of the system is described by the vector $\mathbf{x} = (x_1, \ldots, x_K)$, where element $x_k$ is the number of class-$k$ calls present in the network.

The set of allowed states $\mathcal{S}$ can be described as

$$\mathcal{S} = \{\mathbf{x} : \mathbf{b}_j \cdot \mathbf{x} \leq C_j,\ j = 1, \ldots, J\} \ ,$$

where the scalar product is defined, as usual, as $\mathbf{b}_j \cdot \mathbf{x} = \sum_i b_{j,i} x_i$.

This system has the well known product form stationary distribution

$$\pi(\mathbf{x}) = \frac{1}{G} \prod_{k=1}^{K} \frac{\rho_k^{x_k}}{x_k!} = \frac{1}{G} \prod_{k=1}^{K} f_k(x_k) = \frac{f(\mathbf{x})}{G} \ , \qquad (1)$$

where $f_k(x_k) = \rho_k^{x_k}/x_k!$, and $f(\mathbf{x})$ denotes the unnormalized state probability and $G$ is the so called normalization constant

$$G = \sum_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x}) \ . \qquad (2)$$

The set of blocking states for a class-$k$ call, $\mathcal{B}^k$, is

$$\mathcal{B}^k = \{\mathbf{x} : C_j - \mathbf{b}_j \cdot \mathbf{e}_k < \mathbf{b}_j \cdot \mathbf{x} \leq C_j,\ j = 1, \ldots, J\} \ ,$$

where $\mathbf{e}_k$ is a $K$ component vector with 1 in the $k^{th}$ component and zeros elsewhere. The blocking probability of a class-$k$ call, $B_k$, is then

$$B_k = \sum_{\mathbf{x} \in \mathcal{B}^k} \pi(\mathbf{x}) = \sum_{\mathbf{x} \in \mathcal{S}} \pi(\mathbf{x}) 1_{\mathbf{x} \in \mathcal{B}^k} \ . \qquad (3)$$

In the remainder of this paper we will be dealing with efficient simulation methods for calculating the blocking probabilities as defined in this model.

## 3 Gibbs Sampling for Loss Systems

Our problem is now of the following type. We want to evaluate some quantity $H$ defined as a sum over the allowed state space $\mathcal{S}$,

$$H = \sum_{\mathbf{x} \in \mathcal{S}} h(\mathbf{x}) \ . \qquad (4)$$

In general, the MC method solves the problem by generating i.i.d. samples $\mathbf{X}_n \in \mathcal{S}$ with some distribution $p(\mathbf{x}) = \Pr[\mathbf{X}_n = \mathbf{x}]$ such that $p(\mathbf{x}) \neq 0,\ \forall \mathbf{x} \in \mathcal{S}$. With respect to this distribution $H$ can be written as an expectation

$$H = \sum_{\mathbf{x}\in\mathcal{S}} \frac{h(\mathbf{x})}{p(\mathbf{x})} p(\mathbf{x}) = \mathrm{E}_p\left[h(\mathbf{X})/p(\mathbf{X})\right] \ .$$

The estimator for $H$ when $N$ samples have been drawn is

$$\hat{H} = \frac{1}{N} \sum_{n=1}^{N} \frac{h(\mathbf{X}_n)}{p(\mathbf{X}_n)} \ . \tag{5}$$

Estimator (5) has the correct expectation when each $\mathbf{X}_n$ has the distribution $\pi$, irrespective of whether the $\mathbf{X}_n$ are independent or not. Positive correlation between the samples, however, makes the estimator less efficient from the point of view of its variance.

In our case we are interested in calculating the blocking probabilities as given by eq. (3) with $h(\mathbf{x}) = \pi(\mathbf{x}) 1_{\mathbf{x}\in\mathcal{B}^k}$. Then a natural choice is to let $p(\mathbf{x}) = \pi(\mathbf{x})$, and we get the estimator

$$\hat{B}_k = \frac{1}{N} \sum_{n=1}^{N} 1_{\mathbf{X}_n \in \mathcal{B}^k} \ . \tag{6}$$

One approach for generating the samples is by Markov chain simulation. This relies on the fact that, assuming the holding times are exponentially distributed, our system itself is defined by a Markov chain, e.g. the embedded discrete time Markov chain (jump chain) associated with the arrival and departure epochs. The points in the full jump chain, when weighted with the expected life time of each state, have the stationary distribution $\pi$ and, as noted above, can be used as samples in the summation of eq. (6) despite the fact that they are not independent. Alternatively, one can pick the subchain consisting of the states prior to an arrival or the subchain consisting of the states preceeded by a departure; both of these subchains directly have the stationary distribution $\pi$.

In MCMC methods the idea is the same – to simulate some Markov chain for constructing the distribution $\pi$. The question is only: are there other Markov chains that have the same steady state distribution $\pi$? The answer is yes and, in fact, many of them [5]. The Gibbs sampler introduced later in this chapter is just one of them, but its properties allow us to exploit the product form solution of the steady state distribution in order to gain significant simulation efficiency increases, as will be discussed in chapter 4.

## 3.1 General Theory

Let $\mathbf{X} = (X_1, \ldots, X_K)$ denote the vector random variable for the state of the loss network with the steady state distribution $\pi(\mathbf{x})$ as in (1). We are now interested in ways of constructing a Markov chain $\mathbf{X}_n^*$ having the invariant distribution $\pi$. One way is to use transition probabilities based on conditioning, as defined in the following theorem (taken with slight modification from [5]).

**Theorem 1:** Let sets $\mathcal{A}_1, \ldots, \mathcal{A}_I$ form a partition of the state space $\mathcal{S}$ and let $i(\mathbf{x})$ denote the index of the set to which a state $\mathbf{x}$ in $\mathcal{S}$ belongs to. Let $\mathbf{X}$ be a random variable with distribution $\pi$. The Markov chain $\mathbf{X}_n^*$ with the transition probability

$$\Pr\left[\mathbf{X}_{n+1}^* = \mathbf{y} \mid \mathbf{X}_n^* = \mathbf{x}\right] = \Pr\left[\mathbf{X} = \mathbf{y} \mid \mathbf{X} \in A_{i(\mathbf{x})}\right] \tag{7}$$

has the invariant distribution $\pi$.

**Proof:**

$$\begin{aligned}
\Pr\left[\mathbf{X}_{n+1}^* = \mathbf{y}\right] &= \sum_{\mathbf{x}\in\mathcal{S}} \Pr\left[\mathbf{X}_{n+1}^* = \mathbf{y} \mid \mathbf{X}_n^* = \mathbf{x}\right] \Pr\left[\mathbf{X}_n^* = \mathbf{x}\right] \\
&= \sum_i \sum_{\mathbf{x}\in\mathcal{A}_i} \Pr\left[\mathbf{X}_{n+1}^* = \mathbf{y} \mid \mathbf{X}_n^* = \mathbf{x}\right] \Pr\left[\mathbf{X}_n^* = \mathbf{x}\right] \\
&= \sum_i \sum_{\mathbf{x}\in\mathcal{A}_i} \Pr\left[\mathbf{X} = \mathbf{y} \mid \mathbf{X} \in \mathcal{A}_i\right] \Pr\left[\mathbf{X}_n^* = \mathbf{x}\right] \\
&= \sum_i \Pr\left[\mathbf{X} = \mathbf{y} \mid \mathbf{X} \in \mathcal{A}_i\right] \sum_{\mathbf{x}\in\mathcal{A}_i} \Pr\left[\mathbf{X}_n^* = \mathbf{x}\right] \\
&= \sum_i \Pr\left[\mathbf{X} = \mathbf{y} \mid \mathbf{X} \in \mathcal{A}_i\right] \Pr\left[\mathbf{X}_n^* \in \mathcal{A}_i\right] .
\end{aligned}$$

If $\mathbf{X}_n^*$ has the distribution $\pi$, so does $\mathbf{X}_{n+1}^*$ because then

$$\Pr\left[\mathbf{X}_{n+1}^* = \mathbf{y}\right] = \sum_i \Pr\left[\mathbf{X} = \mathbf{y} \mid \mathbf{X} \in \mathcal{A}_i\right] \Pr\left[\mathbf{X} \in \mathcal{A}_i\right] = \Pr\left[\mathbf{X} = \mathbf{y}\right] = \pi(\mathbf{y}) \ \square$$

Let $P^{(1)}$ denote the probability matrix with transition probabilities as in eq. (7). The Markov chain produced by this is not irreducible, because there are no transitions between different sets. However, by defining several partitions $1, \ldots, M$ we can construct an irreducible Markov chain $\mathbf{X}_n^*$. Let $P^{(m)}$, $m = 1, \ldots, M$, denote the corresponding transition matrices. Then, with a suitable choice of the partitions, the Markov chain $\mathbf{X}_n^*$ corresponding to the compound transition matrix $P = P^{(1)} \cdots P^{(M)}$ will be irreducible. Since each $P^{(m)}$ has the invariant distribution $\pi$ also the compound matrix $P$ will have the invariant distribution $\pi$, and because $\mathbf{X}_n^*$ is now irreducible, $\pi$ is also its unique stationary distribution.

## 3.2 Gibbs Sampler and its Application to Loss Networks

In our case we have a product form solution $\pi$ and it is natural to define the sets in a partition to consist of points in coordinate directions. This leads to the so called Gibbs sampler.

We define $K$ partitions. The sets in partition $k$, $k = 1, \ldots, K$, are denoted with $\mathcal{A}_i^k$. As before, the index of the set in partition $k$ to which a state $\mathbf{x}$ belongs is defined as $i^k(\mathbf{x})$. This set consists of the states

$$\mathcal{A}_{i^k(\mathbf{x})}^k = \{\mathbf{y} = (x_1, \ldots, x_{k-1}, l, x_{k+1}, \ldots, x_K) : l \in \mathcal{I} \ , \mathbf{b}_j \cdot \mathbf{y} \le C_j \ , \ j = 1, \ldots, J\} \ ,$$

where $\mathcal{I}$ is the set of non-negative integers. For the sequel, we denote by $L^k(\mathbf{x})$ the largest value of the component $k$ (variable $l$) of the points in the above set $\mathcal{A}_{i^k(\mathbf{x})}^k$.

To illustrate the concept consider the simple network of Fig. 1, with the state space depicted in the same figure.

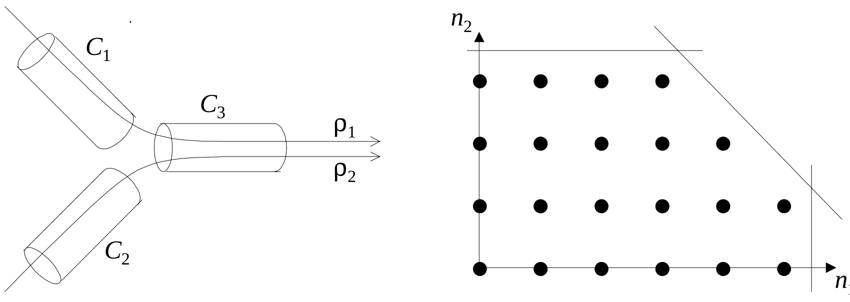

Figure 1: Example network and its state space.

In this case we have two traffic classes, $K = 2$, and we use two different partitions with the "rows" corresponding to partition 1 and the "columns" to partition 2 (see Fig. 2).

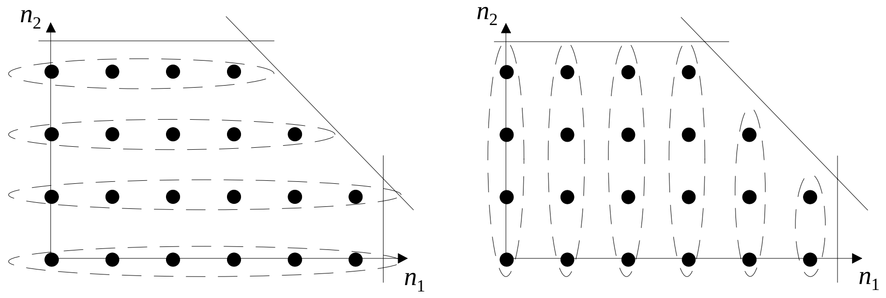

Figure 2: The state space partitions 1 and 2 (left and right).

Associated with each partition, there is a transition matrix $P^{(k)}$ with the transition probabilities (7). Then we construct a compound transition matrix $P = P^{(1)} \cdots P^{(K)}$. The corresponding Markov chain $\mathbf{X}_n^*$ is irreducible since it is possible to move from any state $\mathbf{x}$ in the coordinate convex state space $\mathcal{S}$ to any other state $\mathbf{y}$ with at most $K$ transitions.

The simulation of the Markov chain $\mathbf{X}_n^*$ consists of making transitions with the transition matrices $P^{(k)}$ in cyclical order. In transitions generated with $P^{(k)}$, the state remains in one of the sets $\mathcal{A}^k_{i^k(\mathbf{X}^*_n)}$, i.e. only the component $x_k$ changes. Starting from the state $\mathbf{X}^*_n$ the value of $x_k$ of the next state is obtained by drawing it from the distribution $f_k(x_k)/G_k(L^k(\mathbf{X}^*_n))$, where the normalizing function $G_k(\cdot)$ is defined by

$$G_k(L) = \sum_{l=0}^{L} f_k(l) \ .$$

Along the generated path, we collect information about the number of visits to the blocking states of class-$k$ calls in order to form the estimator (6), i.e.

$$\hat{B}_k = \frac{1}{N} \sum_{n=1}^{N} 1_{\mathbf{X}^*_n \in \mathcal{B}^k} \ .$$

Figure (3) shows a sample path for the previously presented example network. The circled state is the initial state and the six transitions correspond to three full cycles, i.e. three series of transitions in each dimension. As can be seen, the path consists of transitions in the different coordinate directions of the state space. Also, two blocking states of traffic class 2 are visited: states $(2, 3)$ and $(3, 3)$.

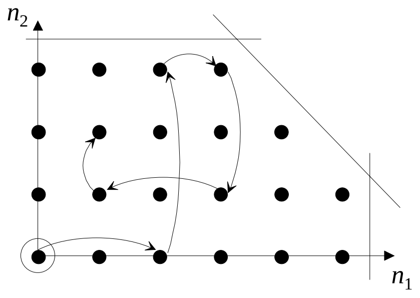

Figure 3: MCMC simulation example

How does this technique differ from the Monte Carlo summation or embedded discrete time Markov chain techniques? In the traditional (or most straight forward) MC summation technique, i.i.d. samples are generated into the state space limited by the maximum number of allowed class-$k$ calls, and the samples that fall outside the real allowed state space $\mathcal{S}$ are rejected. The Gibbs sampler provides a way of generating Monte Carlo samples from the state space $\mathcal{S}$, which is simple requiring only the generation of random variables from univariate truncated Poisson distributions for each transition. The advantage it has is that it manages to eliminate the problem of generating "misses" from the state space $\mathcal{S}$. On the other hand the generation of transitions from the Markov chain of the Gibbs sampler is almost as easy as for generating them from the embedded Markov chain associated with the arrival and departure epochs of the process. The samples generated from the MCMC method are, however, much less correlated than the samples from the embedded Markov chain.

## 3.3 Uniform Sampling with Gibbs Sampler

This idea of partitioning the state space can also be used to generate a uniform distribution over the complete allowed state space $\mathcal{S}$. In general, if we want to take uniformly distributed samples from the state space we will have $p(\mathbf{x}) = 1/S$ in estimator (5), where $S$ denotes the size of the state space $\mathcal{S}$. The uniform distribution is trivially of the product form and the Gibbs sampler is applicable. Now starting from the state $\mathbf{X}_n^*$ the transitions generated with $P^{(k)}$ are obtained by drawing $x_k$ of the next state from the uniform distribution in the range $0, \ldots, L^k(\mathbf{X}_n^*)$. The estimator from (5) for the blocking probability of a class-$k$ call is then

$$\begin{aligned} \hat{B}_k &= \frac{S}{N}\sum_{n=1}^{N} \pi(\mathbf{X}_n^*) 1_{\mathbf{X}_n^* \in \mathcal{B}^k} \\ &= \frac{S}{NG}\sum_{n=1}^{N} f(\mathbf{X}_n^*) 1_{\mathbf{X}_n^* \in \mathcal{B}^k} \ . \end{aligned} \tag{8}$$

Now, $G$ is unknown in (8), but similarly from (2) and (5) we can estimate $G$ by

$$\hat{G} = \frac{S}{N}\sum_{n=1}^{N} f(\mathbf{X}_n^*) \ . \tag{9}$$

Using this in (8) we get another estimator

$$\hat{B}_k = \frac{\sum_{n=1}^{N} f(\mathbf{X}_n^*) 1_{\mathbf{X}_n^* \in \mathcal{B}^k}}{\sum_{n=1}^{N} f(\mathbf{X}_n^*)} = \frac{\hat{G}_k^B}{\hat{G}} \ , \tag{10}$$

where $\hat{G}_k^B$ is the estimator for $G_k^B = \sum_{\mathbf{x} \in \mathcal{B}^k} f(\mathbf{x})$.

This ratio estimator is no longer unbiased. It is strongly consistent though, i.e. $\mathrm{E}[\hat{B}_k] \to B_k,\ I \to \infty$, since both $\hat{G} \to G$ and $\hat{G}_k^B \to G_k^B$ with probability 1 as $I \to \infty$ by the law of large numbers. Then the ratio of $\hat{G}_k^B/\hat{G}$ approaches $G_k^B/G$ as $I \to \infty$.

The uniform sampling may not be a very effective way to do the sampling, because it often "wastes" time on sampling every part of the state space with equal probability, when some parts of the state space are actually more important than the others. In particular, in our case when the distribution $\pi(\mathbf{x})$ is concentrated in a small part of the state space $\mathcal{S}$, the uniform sampling does not necessarily produce very good results.

# 4 Improved Gibbs Sampling Method

The method as described in the previous chapter does not yet give any significant improvement over the known techniques. We can, however, improve the efficiency of the method considerably by utilizing a priori knowledge about the conditional distributions of the sets.

## 4.1 Improved Poisson Sampling

The idea is simple: At each step of the chain, starting from the current state $\mathbf{X}_n^*$, we make a "virtual" transition with transition matrix $P^{(k)}$ to the state $\mathbf{Y}_n^*$. Since in stationary state $\mathbf{X}_n^*$ has the distribution $\pi$ and $P^{(k)}$ has the invariance property, the state $\mathbf{Y}_n^*$ after the virtual transition also has the distribution $\pi$.

We can then use $\mathbf{Y}_n^*$ as a sample point instead of $\mathbf{X}_n^*$ and get the estimator

$$\hat{B}_k = \frac{1}{N} \sum_{n=1}^{N} 1_{\mathbf{Y}_n^* \in \mathcal{B}^k} \ . \tag{11}$$

Now it is possible to take into account the effect of the extra step analytically by calculating the expectation of (11) conditioned on the $\mathbf{X}_n^*$ (see Appendix for a formal explanation on this step). This is easy to do since

$$\mathrm{E}\left[1_{\mathbf{Y}_n^* \in \mathcal{B}^k} \mid \mathbf{X}_n^*\right] = \frac{f_k(L_n^k)}{G_k(L_n^k)} \ ,$$

where $L_n^k = L^k(\mathbf{X}_n^*)$ is the largest value of $x_k$ in the set $\mathcal{A}^k_{i^k(\mathbf{X}_n^*)}$. Then our estimator becomes

$$\hat{B}_k = \frac{1}{N} \sum_{n=1}^{N} \frac{f_k(L_n^k)}{G_k(L_n^k)} \ . \tag{12}$$

In effect, by this method we have included transitions to all states in $\mathcal{A}^k_{i^k(\mathbf{X}_n^*)}$. In particular, a contribution from a blocking state is obtained for every point $\mathbf{X}_n^*$ in the Markov chain. Furthermore, note that the function $f_k(\cdot)/G_k(\cdot)$ can be precomputed for all $k$ and all required values of the argument. Note also that our improvement method does not require the use of the Gibbs sampler to generate the samples $\mathbf{X}_n^*$. Actually, they can be generated by any means provided that the $\mathbf{X}_n^*$ have the distribution $\pi$. Thus using traditional MC summation techniques to generate the $\mathbf{X}_n^*$ is also possible.

In Fig. 4 we have the same example as in the previous chapter. The figure on the left indicates the sets that would be covered by this particular realization and the figure on the right indicates the order in which the different blocking states would be sampled along the simulation. The simulation starts from the state $(0, 0)$ and moves to state $(2, 0)$. Blocking state $(5,0)$ will then be sampled first for class-1 and state $(2,3)$ as the second sample for class-2. Then the process moves to state $(2,3)$. Now state $(3,4)$ will be the third sample for a blocking state for class-1 and state $(2,3)$ will be the fourth sample for the blocking state for class-2. This completes the first simulation cycle. The figure shows a realization of three complete cycles.

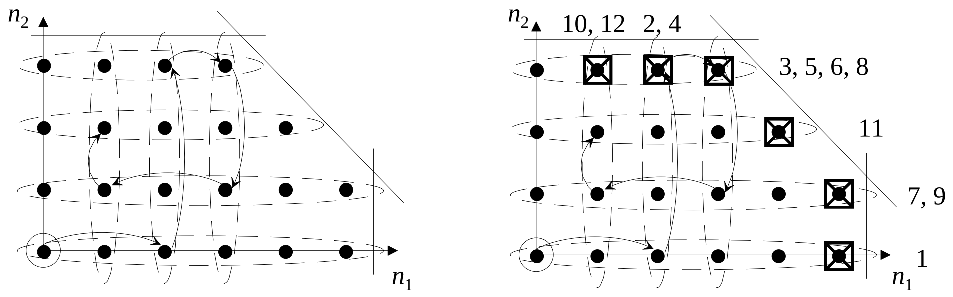

Figure 4: Improved MCMC simulation example

## 4.2 Improved Uniform Sampling

The improvements described in the previous chapter can also be used when sampling the state space with a uniform distribution. Again, let $\mathbf{Y}_n^*$ denote the state after making a virtual transition from the state $\mathbf{X}_n^*$ having the stationary distribution and consider using $\mathbf{Y}_n^*$ for the estimator

$$\hat{B}_k = \frac{1}{GN}\sum_{n=1}^{N} f(\mathbf{Y}_n^*) 1_{\mathbf{Y}_n^* \in \mathcal{B}^k} \ .$$

Calculating analytically the expectation of this extra virtual step conditioned on $\mathbf{X}_n^*$ gives

$$\mathrm{E}\left[f(\mathbf{Y}_n^*) 1_{\mathbf{Y}_n^* \in \mathcal{B}^k} \mid \mathbf{X}_n^*\right] = \frac{1}{L_n^k} f^{(k)}(\mathbf{X}_n^*) f_k(L_n^k) \ ,$$

where $f^{(k)}(\mathbf{X}) = \prod_{l \neq k} f_l(X_l)$, and, as before, $L_n^k = L^k(\mathbf{X}_n^*)$ is the largest value of $x_k$ in the set $\mathcal{A}^k_{i^k(\mathbf{X}_n^*)}$.

The estimator then becomes

$$\hat{B}_k = \frac{1}{GN}\sum_{n=1}^{N} \frac{1}{L_n^k} f^{(k)}(\mathbf{X}_n^*) f_k(L_n^k) \ , \tag{13}$$

where the unknown $G$ can similarly be estimated using the extra "virtual" step,

$$\hat{G} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{L_n^k}f^{(k)}(\mathbf{X}_n^*)\sum_{l=0}^{L_n^k}f_k(l) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{L_n^k}f^{(k)}(\mathbf{X}_n^*)G_k(L_n^k)\ . \tag{14}$$

Combining (13) and (14) gives the following ratio estimator for the blocking probabilities

$$\hat{B}_k = \frac{\sum_{n=1}^{N} f^{(k)}(\mathbf{X}_n^*)f_k(L_n^k)/L_n^k}{\sum_{n=1}^{N} f^{(k)}(\mathbf{X}_n^*)G_k(L_n^k)/L_n^k}\ . \tag{15}$$

Again, the functions $f_k(\cdot)$ and $G_k(\cdot)$ can be separately precalculated and stored into arrays before the actual simulation run.

## 5 Implementation Aspects

When developing methods for simulation one consideration is the computational complexity of the method . Here we show that the method is actually very simple to implement and does not consume memory excessively either. The following issues determine the efficiency of our method:

1. To collect the samples, when considering the "virtual step" from $P^{(k)}$, we must evaluate $f_k(L_n^k)/G_k(L_n^k)$. As mentioned before, the values of this ratio can be calculated in advance and stored into arrays. This requires for each class-$k$ one array of length $N_k^{max}$, denoting the maximum number of allowed class-$k$ calls. For uniform sampling we must store two arrays: one containing the $f_k(L_n^k)$ values and one containing the sums $G_k(L_n^k)$. Thus we note that, actually, the use of the extra analytical step does not increase the computational complexity of the simulation at all.

2. To increase efficiency we note that at each state $\mathbf{X}_n^*$ we can do the "virtual step" for all directions, $k = 1, \ldots, K$, of the system.

3. To generate the actual transitions from $P^{(k)}$ in the Poisson sampling case we need to generate random variables having the truncated Poisson distribution in the range $0, \ldots, L_n^k$ with the offered load $\rho_k$. For this we also need one truncated Poisson distribution for each traffic class-$k$ with maximum length $N_k^{max}$. In the implementation of the algorithm several techniques can then be used for increasing the look-up speed using appropriate data structures. Note that the generation of the transitions in the uniform sampling case is easier, i.e. we only need to generate a uniformly distributed random variable over the range $0, \ldots, L_n^k$, but on the other hand we must calculate an extra term $f^{(k)}(\mathbf{X}_n^*)$ for each sample.

## 6 Numerical Results

Our numerical example consists of the star network studied by Ross [4, p. 240] for the moderate traffic case and we compare our results to the results obtained using the importance sampling heuristics in [4, chap. 6]. The network is a star network with 4 links of capacity around 100 units and there are two service types requiring 1 or 5 bandwidth units. Routes are established between all the leaves of the network for both traffic classes giving a total of 12 classes. The load has been dimensioned such that the blocking probabilities are around $10^{-5} \ldots 10^{-2}$.

The table below shows the results we obtained from using the improved Poisson sampling and uniform sampling and the results of Ross [4, p. 243]. As can be seen our results show very good matching between Ross' results. Note that the results are expressed as percentages.

| Class | Ross' heuristic | Improved MCMC (Poisson) | Improved MCMC (Uniform) |
|---|---|---|---|
| 1 | (0.312, 0.359) | (0.347, 0.350) | (0.333, 0.360) |
| 2 | (0.261, 0.305) | (0.297, 0.300) | (0.288, 0.318) |
| 3 | (0.253, 0.297) | (0.289, 0.292) | (0.273, 0.293) |
| 4 | (0.063, 0.081) | (0.068, 0.069) | (0.067, 0.073) |
| 5 | (0.055, 0.072) | (0.060, 0.061) | (0.060, 0.067) |
| 6 | (0.008, 0.013) | (0.010, 0.011) | (0.010, 0.011) |
| 7 | (2.200, 2.340) | (2.286, 2.296) | (2.190, 2.322) |
| 8 | (1.870, 2.000) | (1.960, 1.969) | (1.906, 2.039) |
| 9 | (1.820, 1.950) | (1.901, 1.920) | (1.855, 1.983) |
| 10 | (0.463, 0.513) | (0.491, 0.494) | (0.483, 0.520) |
| 11 | (0.414, 0.462) | (0.431, 0.434) | (0.434, 0.469) |
| 12 | (0.065, 0.080) | (0.080, 0.081) | (0.081, 0.087) |

However, Ross obtained his results by generating 100000 i.i.d. MC samples and we used 50 independent runs of simulations with 50000 cycles (each roughly corresponding to one MC sample). In order to make the results comparable, we have scaled our results by multiplying the confidence intervals by $\sqrt{25}$. The table below shows the lengths of the confidence intervals for each method after the scaling. As expected, the uniform sampling method does not give particularly good results, but the Poisson sampling method gives confidence intervals approximately half of those of Ross' results indicating clear variance reduction.

| Class | Ross' heuristic | Improved MCMC (Poisson) | Improved MCMC (Uniform) |
|---|---|---|---|
| 1 | 0.047 | 0.016 | 0.134 |
| 2 | 0.044 | 0.016 | 0.153 |
| 3 | 0.044 | 0.016 | 0.104 |
| 4 | 0.018 | 0.006 | 0.030 |
| 5 | 0.017 | 0.006 | 0.033 |
| 6 | 0.005 | 0.002 | 0.005 |
| 7 | 0.140 | 0.050 | 0.659 |
| 8 | 0.130 | 0.043 | 0.663 |
| 9 | 0.130 | 0.052 | 0.639 |
| 10 | 0.050 | 0.018 | 0.183 |
| 11 | 0.048 | 0.018 | 0.174 |
| 12 | 0.015 | 0.006 | 0.031 |

We also made tests on the rare event example of Heegaard [3], but noticed a tendency to underestimate the probabilities. The reason is that when performing rare event simulation the Markov chain of the Gibbs sampler is confined to move only within a very small part of the whole state space. Then it does not necessarily sample the most important blocking states for all traffic classes. Therefore, when dealing with rare event simulation our improved method needs to be combined with importance sampling methods to shift the probability mass from very close to the origin of the state space closer to the state space boundaries.

## 7 Conclusions

In this article we have presented an efficient simulation method for calculating the blocking probabilities for calls in a multiservice loss system. The method is based on the use of the Gibbs sampler with an appropriate partition of the state space. We are then able to exploit the product form solution by calculating analytically the effect of using as samples not the current state $\mathbf{X}_n^*$ of the simulation of the Markov chain, but a "virtual" sample generated from the current state with a transition matrix $P^{(k)}$. It is then possible to calculate analytically the probability of this new sample hitting the blocking state of a class-$k$ call. Thus for each state in the chain we get a contribution from the blocking state for class-$k$ call within the current set. Furthermore, this can be done in each of the $K$ directions of the system giving a blocking sample for each traffic class. The inclusion of this extra analytical step in the simulation gives clear reduction in the variance over the traditional MC summation techniques as shown by our numerical results. Also, it was shown that the improved method does not cause excessive computational complexity.

## Acknowledgement

The authors wish to thank Samuli Aalto for many useful discussions during the course of the work.

## References

[1] M. A. Crane, D. L. Iglehart, “Simulating Stable Stochastic Systems: III. Regenerative Processes and Discrete Event Simulations”, Operations Research vol. 23, no. 1, 1975

[2] M. A. Crane, A. J. Lemoine, “An Introduction to the Regenerative Method for Simulation Analysis”, Springer-Verlag 1977, 111 p.

[3] P. E. Heegaard, “Efficient Simulation of Network Performance by Simulation”, 15th International Teletraffic congress - ITC 15, Washington D.C., USA, 23-27 June, 1997, Session: Simulation, Measurement and Data Analysis

[4] K. W. Ross, “Multiservice Loss Models for Broadband Telecommunication Networks”, Springer-Verlag, 1995, 343 p.

[5] L. Tierney, “Markov Chains for Exploring Posterior Distributions”, The Annals of Statistics 1994, vol. 22, No. 4, p. 1701-1762

## Appendix

Consider the sum

$$H = \sum_{\mathbf{x} \in \mathcal{S}} h(\mathbf{x}) p(\mathbf{x}) = \mathrm{E}\left[h(\mathbf{X})\right] \ ,$$

where $\mathbf{X}$ is a random variable with the distribution $p(\mathbf{x})$, and the corresponding estimator

$$\hat{H} = \frac{1}{N} \sum_{n=1}^{N} h(\mathbf{X}_n) \ ,$$

where $\mathbf{X}_n$ are samples of the random variable $\mathbf{X}$.

Let $P$ be any transition matrix with invariant distribution $p(\mathbf{x})$. Then $\mathbf{Y}$ which is obtained from $\mathbf{X}$ with this matrix also has the distribution $p(\mathbf{x})$. $H$ may then be written as

$$\begin{aligned} H &= \mathrm{E}\left[h(\mathbf{Y})\right] \\ &= \mathrm{E}\left[\mathrm{E}\left[h(\mathbf{Y}) \mid \mathbf{X}\right]\right] \ . \end{aligned}$$

Assume now that $\mathrm{E}[h(\mathbf{Y}) \mid \mathbf{X}]$ can be calculated analytically. Then a new estimator for $H$ can be written

$$\hat{H} = \frac{1}{N} \sum_{n=1}^{N} \mathrm{E}\left[h(\mathbf{Y}) \mid \mathbf{X} = \mathbf{X}_n\right] \ ,$$

which for each sample $\mathbf{X}_n$ includes all the points reachable from $\mathbf{X}_n$ with the transition matrix $P$.